Fisheries-in-R.R

2022-12-11

#R code built around Introductory Fisheries Analysis with R  
#Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.  
  
################################################  
########Day 2: Fisheries Analysis in R##########  
################################################  
  
  
####Topics################################  
# Age-length keys  
# Size structure  
# Weight-length relationships  
# Condition factors  
# Mortality  
# Individual growth  
# Stock-Recruitment  
# Population Estimates  
#########################################  
  
#Load packages for today  
library(FSA)

## ## FSA v0.9.3. See citation('FSA') if used in publication.  
## ## Run fishR() for related website and fishR('IFAR') for related book.

library(FSAdata)

## ## FSAdata v0.3.9. See ?FSAdata to find data for specific fisheries analyses.

library(nlstools) #needed for calculating 95% confidence intervals

##   
## 'nlstools' has been loaded.

## IMPORTANT NOTICE: Most nonlinear regression models and data set examples

## related to predictive microbiolgy have been moved to the package 'nlsMicrobio'

library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

#Age-length keys#######  
  
#Load a bass data set from Florida, name includes an upper case "O" not zero  
RockBassLO2 <- FSAdata::RockBassLO2  
  
#Add length intervals  
#Requires column with lengths, the data frame name, a numeric value   
#identifying the starting length measurement and the width of the length   
#measurement category  
RockBassLO2 <- lencat(~tl,data=RockBassLO2,  
 startcat=110,w=10)  
  
#Data contains missing age values, NA. We want to assign ages based on the ALK.   
  
#First create an "age" data frame with all complete records  
rb.age <- RockBassLO2 %>%  
 dplyr::filter( !is.na(age) )  
  
head(rb.age)

## age tl LCat  
## 1 6 218 210  
## 2 5 184 180  
## 3 7 211 210  
## 4 9 223 220  
## 5 9 245 240  
## 6 7 181 180

#then create a "length" data frame with records with missing ages.  
#Note the only difference from above is the !  
rb.len <- RockBassLO2 %>%  
 dplyr::filter( is.na(age) )  
  
head(rb.len)

## age tl LCat  
## 1 NA 172 170  
## 2 NA 173 170  
## 3 NA 175 170  
## 4 NA 171 170  
## 5 NA 173 170  
## 6 NA 184 180

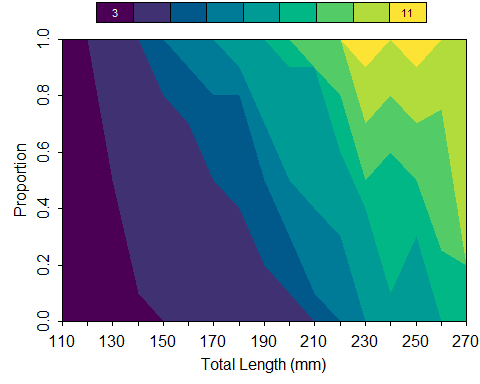
#Construct the age-length key  
  
#Use the xtabs() function to construct a contingency table of the number of   
#fish in each length and age category. The row variable is the first argument   
#and the column variable is the second argument  
rb.raw <- xtabs(~LCat+age,data=rb.age)  
rb.raw

## age  
## LCat 3 4 5 6 7 8 9 10 11  
## 110 1 0 0 0 0 0 0 0 0  
## 120 1 0 0 0 0 0 0 0 0  
## 130 2 2 0 0 0 0 0 0 0  
## 140 1 9 0 0 0 0 0 0 0  
## 150 0 8 2 0 0 0 0 0 0  
## 160 0 7 2 1 0 0 0 0 0  
## 170 0 5 3 2 0 0 0 0 0  
## 180 0 4 4 1 1 0 0 0 0  
## 190 0 2 3 2 3 0 0 0 0  
## 200 0 1 2 2 4 1 0 0 0  
## 210 0 0 1 3 5 0 1 0 0  
## 220 0 0 0 3 3 2 2 0 0  
## 230 0 0 0 0 4 1 2 2 1  
## 240 0 0 0 0 1 5 2 2 0  
## 250 0 0 0 0 3 2 2 2 1  
## 260 0 0 0 0 0 1 2 1 0  
## 270 0 0 0 0 0 1 0 4 0

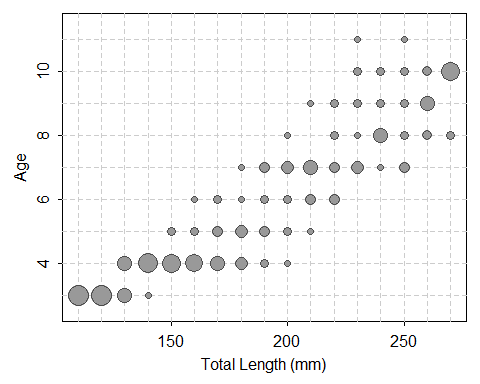
#convert counts to proportions using the prop.table() function. Requires the   
#table created above and margin=1.  
#margin=1 tells R to calculate proportions by row  
#margin=2 tells R to calculate proportions by column  
rb.key <- prop.table(rb.raw,margin=1)  
rb.key

## age  
## LCat 3 4 5 6 7 8 9 10 11  
## 110 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 120 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 130 0.50 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 140 0.10 0.90 0.00 0.00 0.00 0.00 0.00 0.00 0.00  
## 150 0.00 0.80 0.20 0.00 0.00 0.00 0.00 0.00 0.00  
## 160 0.00 0.70 0.20 0.10 0.00 0.00 0.00 0.00 0.00  
## 170 0.00 0.50 0.30 0.20 0.00 0.00 0.00 0.00 0.00  
## 180 0.00 0.40 0.40 0.10 0.10 0.00 0.00 0.00 0.00  
## 190 0.00 0.20 0.30 0.20 0.30 0.00 0.00 0.00 0.00  
## 200 0.00 0.10 0.20 0.20 0.40 0.10 0.00 0.00 0.00  
## 210 0.00 0.00 0.10 0.30 0.50 0.00 0.10 0.00 0.00  
## 220 0.00 0.00 0.00 0.30 0.30 0.20 0.20 0.00 0.00  
## 230 0.00 0.00 0.00 0.00 0.40 0.10 0.20 0.20 0.10  
## 240 0.00 0.00 0.00 0.00 0.10 0.50 0.20 0.20 0.00  
## 250 0.00 0.00 0.00 0.00 0.30 0.20 0.20 0.20 0.10  
## 260 0.00 0.00 0.00 0.00 0.00 0.25 0.50 0.25 0.00  
## 270 0.00 0.00 0.00 0.00 0.00 0.20 0.00 0.80 0.00

#Visualizing Age-Length key  
alkPlot(rb.key, type = "area", showLegend = TRUE,  
 leg.cex = 0.7, xlab = "Total Length (mm)")



#Bubble size is proportional to the number of fish   
#in each length interval  
alkPlot(rb.key, type = "bubble",   
 xlab = "Total Length (mm)")



#now we are ready to assign ages to individuals without ages. This example   
#uses a semi-random method of assigning ages to individuals  
#Suppose there are 20 fish in a length category that need assigned an age and  
#the age-length key says that 75% in the length category are age-6 and 25% are age-7.   
#The age assignment for age-6 is (0.75 \* 20) = 15  
#The age assignment for age-7 is (0.25 \* 20) = 5  
  
#What about the fractional assignments for 22 fish?  
#(0.75 \* 22) = 16.5  
#(0.25 \* 22) = 5.5  
#FSA rounds down so that 16 fish are assigned age-6 and 5 fish are assigned age-7.  
  
#The remaining fish is assigned to age-6 with a   
#probability of 0.75 or age-7 with a probability of 0.25  
  
rb.len1 <- alkIndivAge(rb.key, age ~ tl,   
 data = rb.len,   
 type = c("SR"))  
  
#Combine aged and unaged (but with new ages) samples  
rb.combined = rbind(rb.age, rb.len1)  
  
# Calculate mean length-at-age assuming fully   
#random selection  
Summarize(tl~age,data=rb.combined,digits=2)

## age n mean sd min Q1 median Q3 max  
## 1 3 5 129.40 12.28 111 125.00 131.0 137.00 143  
## 2 4 319 175.73 15.58 130 164.00 176.0 186.50 208  
## 3 5 264 187.16 14.29 151 177.75 187.0 197.00 218  
## 4 6 216 200.25 17.15 164 188.00 202.0 214.00 228  
## 5 7 307 209.39 15.49 181 198.00 207.0 217.00 258  
## 6 8 77 226.90 17.56 200 207.00 225.0 241.00 270  
## 7 9 64 228.62 13.70 210 221.00 225.5 235.75 265  
## 8 10 30 246.10 14.16 231 236.00 242.5 252.50 278  
## 9 11 6 238.33 9.93 230 234.25 235.5 236.75 258

# age frequency distribution  
af <- xtabs(~age,data=rb.combined)  
# proportional age distribution  
ap <- prop.table(af)  
ap

## age  
## 3 4 5 6 7 8   
## 0.003881988 0.247670807 0.204968944 0.167701863 0.238354037 0.059782609   
## 9 10 11   
## 0.049689441 0.023291925 0.004658385

#Calculate mean length-at-age following Bettoli and Miranda 2001.   
#Used when fish are aged with a stratified design to get number of   
#fish in each length interval in the entire sample.  
  
#Generate number per length class  
len.n <- xtabs(~LCat,data=RockBassLO2)  
  
#Calculate mean length at age and SD  
alkMeanVar(rb.key, tl ~ LCat + age,   
 data = rb.age,   
 len.n = len.n)

## age mean sd  
## 1 3 129.6667 12.13072  
## 2 4 177.0340 16.05173  
## 3 5 187.2448 13.77540  
## 4 6 200.1241 17.26455  
## 5 7 208.3250 15.40027  
## 6 8 227.2428 16.58362  
## 7 9 227.4819 14.56039  
## 8 10 246.0548 14.45170  
## 9 11 241.3590 10.11402

#Size structure#######  
  
#Load Largemouth Bass data from the FSAdata package  
LMBassBL <- FSAdata::LMBassBL   
  
#Add a length category column  
LMBassBL$lcat10 <- lencat(LMBassBL$tl,w=10)  
  
#Check data to confirm  
headtail(LMBassBL)

## tl lcat10  
## 1 87 80  
## 2 90 90  
## 3 99 90  
## 445 499 490  
## 446 504 500  
## 447 510 510

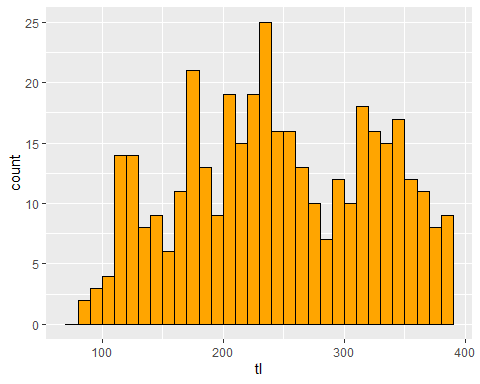
#Create a length frequency table  
#Using the xtabs function, requires the column name   
#with length category and the dataset  
LMBFreq10 <- xtabs(~lcat10, data = LMBassBL)  
#print frequency table  
LMBFreq10

## lcat10  
## 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270   
## 1 4 4 11 16 9 8 6 11 20 13 10 17 15 20 25 18 14 13 12   
## 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470   
## 5 12 11 17 15 16 18 11 11 9 10 13 11 7 9 5 7 5 3 1   
## 490 500 510   
## 2 1 1

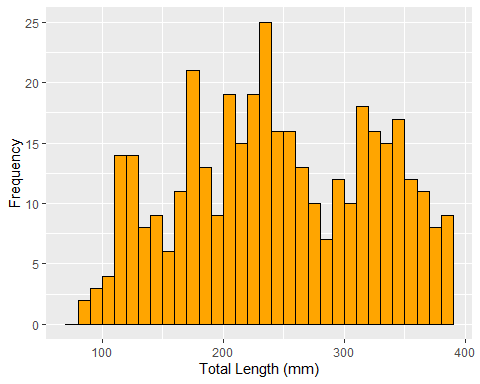
#Percentage of fish in each interval using the   
#prop.table() function  
LMBPer10 <- round( prop.table(LMBFreq10) \* 100 , 1)  
#print proportion table  
LMBPer10

## lcat10  
## 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270   
## 0.2 0.9 0.9 2.5 3.6 2.0 1.8 1.3 2.5 4.5 2.9 2.2 3.8 3.4 4.5 5.6 4.0 3.1 2.9 2.7   
## 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470   
## 1.1 2.7 2.5 3.8 3.4 3.6 4.0 2.5 2.5 2.0 2.2 2.9 2.5 1.6 2.0 1.1 1.6 1.1 0.7 0.2   
## 490 500 510   
## 0.4 0.2 0.2

#A histogram can be useful to visualize   
#length frequencies  
ggplot(LMBassBL, aes(x = tl)) +   
 geom\_histogram(breaks = seq(from = 70 , to = 390, by = 10),  
 fill = "orange", color = "black")



###Histogram practice#############  
#Add "Total Length (mm)" to the x-axis label  
#Add "Frequency" to the y-axis label  
  
###Answer################  
ggplot(LMBassBL, aes(x = tl)) +   
 geom\_histogram(breaks = seq(from = 70 , to = 390,   
 by = 10),  
 fill = "orange", color = "black") +  
 xlab("Total Length (mm)") +  
 ylab("Frequency")



###End Answer###########  
  
  
###PSD####  
#FSA contains a list of Gabelhouse (1984)   
#length categories  
psdVal("Largemouth Bass")

## substock stock quality preferred memorable trophy   
## 0 200 300 380 510 630

#Use psdVal() to return list of available species  
psdVal()

##   
## Species name must be one of following. Be careful of spelling and capitalization.

## [1] "Arctic Grayling" "Bighead Carp"   
## [3] "Bigmouth Buffalo" "Black Bullhead"   
## Omitted remaining lines

#The units returned are in mm but you can request   
#cm, mm, or in  
psdVal("Largemouth Bass", units = "cm")

## substock stock quality preferred memorable trophy   
## 0 20 30 38 51 63

psdVal("Largemouth Bass", units = "mm")

## substock stock quality preferred memorable trophy   
## 0 200 300 380 510 630

psdVal("Largemouth Bass", units = "in")

## substock stock quality preferred memorable trophy   
## 0 8 12 15 20 25

#Assign length category to each fish to the Largemouth Bass data frame  
  
#Load the LMBassBL data set  
LMBassBL <- FSAdata::LMBassBL   
  
#First we need to pull out the Largemouth Bass   
#length categories  
lmb.cuts <- psdVal("Largemouth Bass", units = "mm")  
  
#Filter based on tl greater then stock length and add the length category column  
LMB\_SS <- LMBassBL %>%  
 filter( tl >= lmb.cuts["stock"]) %>%  
 mutate( gcat = lencat(tl, breaks = lmb.cuts,  
 use.names = TRUE))  
  
headtail(LMB\_SS)

## tl gcat  
## 1 200 stock  
## 2 204 stock  
## 3 204 stock  
## 332 499 preferred  
## 333 504 preferred  
## 334 510 memorable

#Calculate a frequency table across size groups  
xtabs(~gcat, data = LMB\_SS)

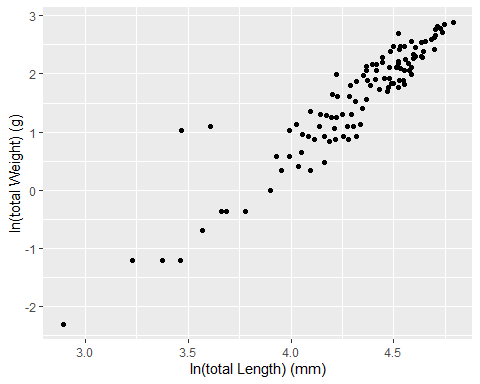
## gcat  
## substock stock quality preferred memorable trophy   
## 0 151 108 74 1 0

#Calculate all PSD-X values and 95% Confidence Intervals. Requires length column, data set,   
#species, and what type. Types available are incremental (e.g., Stock to Quality) or  
#traditional (e.g., Quality, Preferred, Memorable)  
psdCalc(~tl, data=LMB\_SS,   
 species = "Largemouth Bass",   
 what = "traditional")

## Warning: Some category sample size <20, some CI coverage may be  
## lower than 95%.

## Estimate 95% LCI 95% UCI  
## PSD-Q 55 47 62  
## PSD-P 22 16 29

#Break?  
  
#Weight-length relationships#######  
#Load Chinook data from the FSA package  
Chinook <- FSA::ChinookArg  
  
#Plot TL vs TW on the natural log scale  
ggplot(ChinookArg, aes(x = log(tl), y = log(w))) +  
 geom\_point() +  
 xlab("ln(total Length) (mm)") +   
 ylab("ln(total Weight) (g)")



#Estimate parameters of the weight-length model   
#using lm()  
lm1 <- lm(log(w) ~ log(tl), data=ChinookArg)  
  
#Extract summary information and send to a new object  
sumlw <- summary(lm1)  
sumlw

##   
## Call:  
## lm(formula = log(w) ~ log(tl), data = ChinookArg)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.69071 -0.21434 -0.00143 0.17011 1.70527   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -9.86574 0.42048 -23.46 <2e-16 \*\*\*  
## log(tl) 2.64931 0.09707 27.29 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3516 on 110 degrees of freedom  
## Multiple R-squared: 0.8713, Adjusted R-squared: 0.8701   
## F-statistic: 744.8 on 1 and 110 DF, p-value: < 2.2e-16

#Extract coefficients and send to a new object  
coeflw <- coef(lm1)  
coeflw

## (Intercept) log(tl)   
## -9.865737 2.649314

#Extract confidence intervals for coefficients  
confinlw <- confint(lm1)  
confinlw

## 2.5 % 97.5 %  
## (Intercept) -10.699032 -9.032442  
## log(tl) 2.456936 2.841693

#Exercise 3  
  
#Lunch?  
  
  
#Condition factors#######  
#Load Bluegill data from the FSA package  
BLG <- FSAdata::BluegillLM   
#Select the data and calculate log10 of   
#length and weight  
  
BLGSub <- BLG %>%  
 mutate(logW = log10(wght), logL = log10(tl)) %>% #take log10  
 select( -c(sl, fl, sernum)) #remove excess columns  
  
headtail(BLGSub)

## tl wght logW logL  
## 1 157 92 1.963788 2.195900  
## 2 153 88 1.944483 2.184691  
## 3 149 77 1.886491 2.173186  
## 98 147 64 1.806180 2.167317  
## 99 144 64 1.806180 2.158362  
## 100 133 49 1.690196 2.123852

#Fulton's Condition Factor  
#Describes condition of individual fish  
#Metric: K = W / L^3 \* 100,000  
#English: K = W / L^3 \* 10,000  
  
BLGSub <- BLGSub %>%   
 mutate(K = wght / (tl^3) \* 100000)  
  
headtail(BLGSub)

## tl wght logW logL K  
## 1 157 92 1.963788 2.195900 2.377327  
## 2 153 88 1.944483 2.184691 2.457018  
## 3 149 77 1.886491 2.173186 2.327726  
## 98 147 64 1.806180 2.167317 2.014782  
## 99 144 64 1.806180 2.158362 2.143347  
## 100 133 49 1.690196 2.123852 2.082769

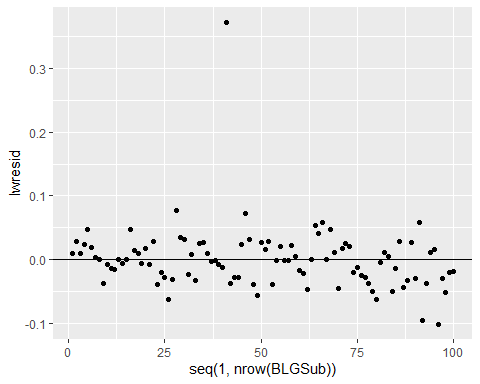
#Weight-Length Residuals  
#Estimate coefficients of weight-length model  
lm1 <- lm(logW ~ logL, data = BLGSub)  
coef(lm1)

## (Intercept) logL   
## -5.524963 3.406255

#Calculate weight residuals  
#Weight residuals are the difference between the observed log10 weight and predicted log10 weight  
#Residuals can tell you if the fish is plumper or skinnier than average.  
  
BLGSub <- BLGSub %>%  
 mutate(lwresid = residuals(lm1))  
  
headtail(BLGSub)

## tl wght logW logL K lwresid  
## 1 157 92 1.963788 2.195900 2.377327 0.008956946  
## 2 153 88 1.944483 2.184691 2.457018 0.027829849  
## 3 149 77 1.886491 2.173186 2.327726 0.009027417  
## 98 147 64 1.806180 2.167317 2.014782 -0.051292251  
## 99 144 64 1.806180 2.158362 2.143347 -0.020789775  
## 100 133 49 1.690196 2.123852 2.082769 -0.019220917

#Plot residuals  
ggplot(BLGSub, aes(x = seq(1,nrow(BLGSub)), y = lwresid)) +  
 geom\_point() +  
 geom\_hline(yintercept=0)



#Relative Weight  
#Wr = W / Ws \* 100 (where Ws is the standard weight given length)  
#Ws = 10 ^ (alpha + beta \* log10(TL))  
  
#FSA contains a list of standard weights from a variety of sources  
#Return list of available species  
wsVal()

##   
## Species name must be one of following. Be careful of spelling and capitalization.

## [1] "Aegean Chub" "African Sharptooth Catfish"   
## [3] "Alabama Bass" "Ankara Nase"   
## Omitted remaining lines

#View the table of coefficients with references  
View(WSlit)  
  
#Return standard weight coefficients for one species  
wsVal("Bluegill", units = c("metric"))

## species units type ref measure method min.TL int slope source  
## 26 Bluegill metric linear 75 TL Other 80 -5.374 3.316 Hillman (1982)

#Return a simplified object for calculation  
wsBlg <- wsVal("Bluegill", units=c("metric"),   
 simplify = TRUE)  
  
#How to reference the intercept and slope  
wsBlg

## species min.TL int slope  
## 26 Bluegill 80 -5.374 3.316

wsBlg$int

## [1] -5.374

wsBlg$slope

## [1] 3.316

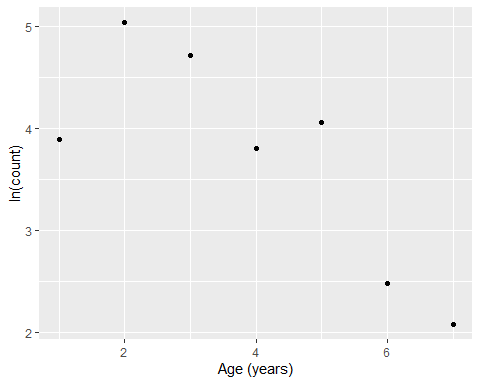
#Add Ws and Wr column  
BLGSub <- BLGSub %>%  
 mutate(Ws = 10 ^(wsBlg$int + wsBlg$slope \* logL ),  
 Wr = wght/Ws \* 100)  
  
headtail(BLGSub)

## tl wght logW logL K lwresid Ws Wr  
## 1 157 92 1.963788 2.195900 2.377327 0.008956946 80.83571 113.81109  
## 2 153 88 1.944483 2.184691 2.457018 0.027829849 74.20562 118.58940  
## 3 149 77 1.886491 2.173186 2.327726 0.009027417 67.96505 113.29352  
## 98 147 64 1.806180 2.167317 2.014782 -0.051292251 64.98667 98.48174  
## 99 144 64 1.806180 2.158362 2.143347 -0.020789775 60.69180 105.45082  
## 100 133 49 1.690196 2.123852 2.082769 -0.019220917 46.63286 105.07612

#Mortality#######  
#This example will calculate instantaneous total mortality (Z)  
  
#Create a data frame for say, Brook Trout  
bkt <- data.frame(age=1:7,  
 ct=c(49,155,112,45,58,12,8))  
bkt

## age ct  
## 1 1 49  
## 2 2 155  
## 3 3 112  
## 4 4 45  
## 5 5 58  
## 6 6 12  
## 7 7 8

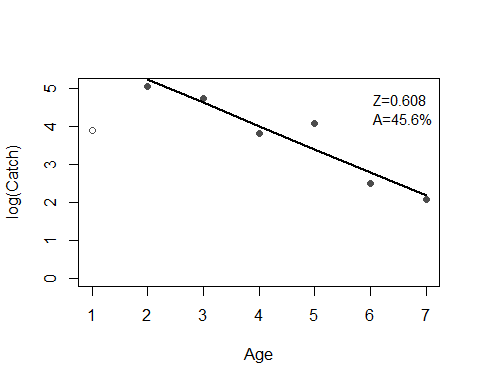
#Create a quick scatterplot with log(ct) to   
#identify the descending limb of the catch curve  
ggplot(bkt, aes(x = age, y = log(ct))) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("ln(count)")



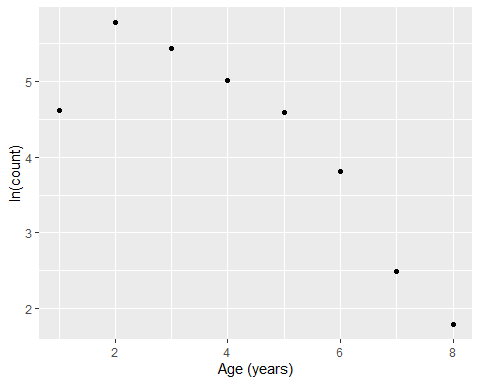
#The catchCurve() function requires  
#1. Formula in the form of catch ~ age  
#2. A data argument, does not have to contain only the descending limb.  
#3. A required age2use argument that specifies the ages to use  
  
bktcc <- catchCurve(ct ~ age, data = bkt, ages2use=2:7)  
  
#The summary function will return  
#the instantaneous mortality (Z)  
#and annual mortality (A)  
summary(bktcc)

## Estimate Std. Error t value Pr(>|t|)  
## Z 0.6076261 0.0944847 6.430947 0.003006749  
## A 45.5357746 NA NA NA

#the plot() function will create a plot of the   
#catch curve  
plot(bktcc)



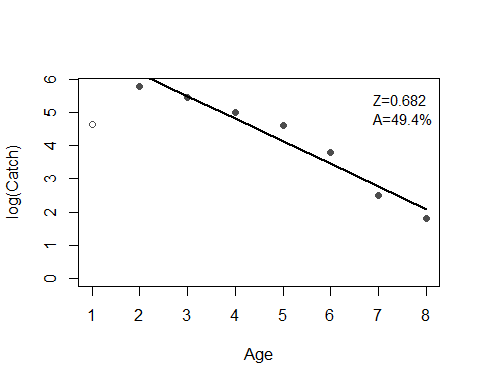
###Mortality Practice#############  
#Calculate Z and A from the following data frame on   
#your own  
lmbcatch <- data.frame(age=1:8,  
 ct=c(102, 325, 230, 150, 99, 45, 12, 6))  
  
###Answer########  
ggplot(lmbcatch, aes(x = age, y = log(ct))) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("ln(count)")



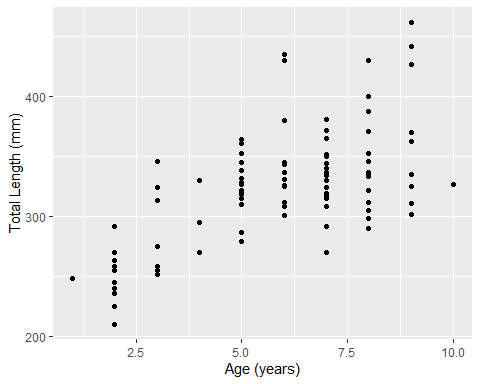
lmbcc <- catchCurve(ct ~ age, data = lmbcatch,   
 ages2use=2:8)  
summary(lmbcc)

## Estimate Std. Error t value Pr(>|t|)  
## Z 0.6816613 0.07100172 9.600631 0.0002077792  
## A 49.4223931 NA NA NA

plot(lmbcc)



###End Answer###########  
  
  
#LVB growth model#######  
#Code for fitting a von Bertalanffy Growth Model  
  
#Load Croaker2 data from the FSAdata package  
Croaker2 <- FSAdata::Croaker2  
  
#Subset to only Males  
crm <- subset(Croaker2, sex=="M")  
  
  
#plot the data to visualize trends  
ggplot(crm, aes(x = age, y = tl)) +  
 geom\_point() +  
 xlab("Age (years)") +   
 ylab("Total Length (mm)")



#Select the von Bertalanffy Growth model to use  
#typical will use the traditional LVB model  
# Linf \* (1 - exp(-K \* (t - t0)))  
vbT <- vbFuns("typical")  
  
  
#Use the non-linear least squares algorithm to   
#estimate parameters. vbT() arguments must be in   
#the order: age, Linf, K, t0 if using "typical"  
fitCroaker <- nls(tl ~ vbT(age, Linf ,K, t0),  
 data=crm,  
 start=vbStarts(tl ~ age,   
 data = crm,   
 type="typical"))  
  
#Extract summary information and send to a new object  
sumCroaker <- summary(fitCroaker, correlation = TRUE)  
sumCroaker

##   
## Formula: tl ~ vbT(age, Linf, K, t0)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Linf 366.4155 16.7550 21.869 < 2e-16 \*\*\*  
## K 0.3148 0.1076 2.924 0.00419 \*\*   
## t0 -1.7143 1.0486 -1.635 0.10493   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 33.42 on 111 degrees of freedom  
##   
## Correlation of Parameter Estimates:  
## Linf K   
## K -0.95   
## t0 -0.87 0.97  
##   
## Number of iterations to convergence: 4   
## Achieved convergence tolerance: 5.398e-06

#Extract coefficients and send to a new object  
coefCroaker <- coef(fitCroaker)  
coefCroaker

## Linf K t0   
## 366.4155241 0.3147927 -1.7142840

#Calculate 95% confidence intervals for coefficients  
confinCroaker <- confint2(fitCroaker)  
confinCroaker

## 2.5 % 97.5 %  
## Linf 333.2143269 399.6167213  
## K 0.1014851 0.5281003  
## t0 -3.7922333 0.3636653

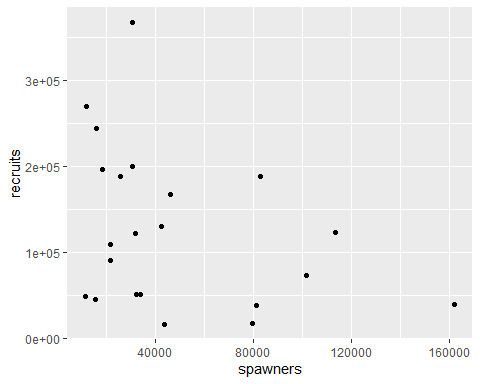
#Common error during parameter estimation, number of iterations exceeded maximum of 50  
  
#Occurs if algorithm has trouble finding coefficients for the model  
#Try increasing the algorithms maximum number of iterations  
fitCroaker <- nls(tl ~ vbT(age, Linf ,K, t0),  
 data=crm,  
 start=vbStarts(tl ~ age,   
 data = crm,   
 type="typical"),  
 control=list(maxiter=1000))  
  
summary(fitCroaker, correlation = TRUE)

##   
## Formula: tl ~ vbT(age, Linf, K, t0)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Linf 366.4155 16.7550 21.869 < 2e-16 \*\*\*  
## K 0.3148 0.1076 2.924 0.00419 \*\*   
## t0 -1.7143 1.0486 -1.635 0.10493   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 33.42 on 111 degrees of freedom  
##   
## Correlation of Parameter Estimates:  
## Linf K   
## K -0.95   
## t0 -0.87 0.97  
##   
## Number of iterations to convergence: 4   
## Achieved convergence tolerance: 5.398e-06

#Break?  
  
#Stock-Recruitment#######  
  
#Stock and recruitment data for Klamath River  
#Chinook salmon, 1979-2000  
ChinookSR <- FSAdata::ChinookKR  
#Remove incomplete records  
ChinookSR <- na.omit(ChinookSR)  
  
headtail(ChinookSR)

## brood.year spawners recruits  
## 1 1979 30637 200698  
## 2 1980 21484 109430  
## 3 1981 33857 50968  
## 20 1998 42488 130283  
## 21 1999 18457 196197  
## 22 2000 82728 188537

#Plot spawners (stock) vs recruits  
ggplot(ChinookSR, aes(x = spawners, y = recruits)) +  
 geom\_point()



#Ricker function  
#E[R|S] = alpha \* S \* exp(-beta \* S)  
  
#We will use the nls function to fit this non-linear model  
#Requires starting values  
svR <- srStarts(recruits ~ spawners,   
 data = ChinookSR,   
 type = "Ricker")  
svR

## $a  
## [1] 8.527721  
##   
## $b  
## [1] 2.517053e-05

#Obtain Ricker function from FSA  
rckr <- srFuns("Ricker")  
  
#Fit Ricker function to stock and recruitment data  
srR <- nls(recruits ~ rckr(spawners,a,b),   
 data = ChinookSR,   
 start=svR)  
summary(srR)

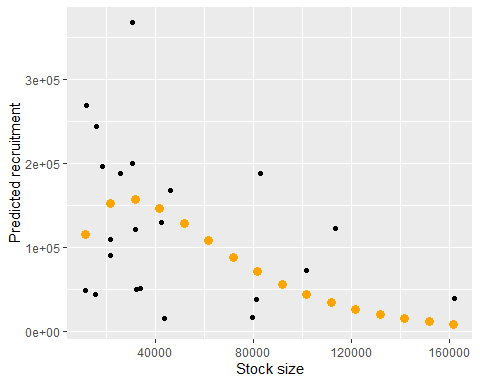
##   
## Formula: recruits ~ rckr(spawners, a, b)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## a 1.494e+01 4.964e+00 3.010 0.00692 \*\*  
## b 3.475e-05 9.481e-06 3.665 0.00154 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 92630 on 20 degrees of freedom  
##   
## Number of iterations to convergence: 11   
## Achieved convergence tolerance: 4.46e-06

#Coefficients with 95% Confidence Intervals  
cbind(estimates=coef(srR), confint(srR))

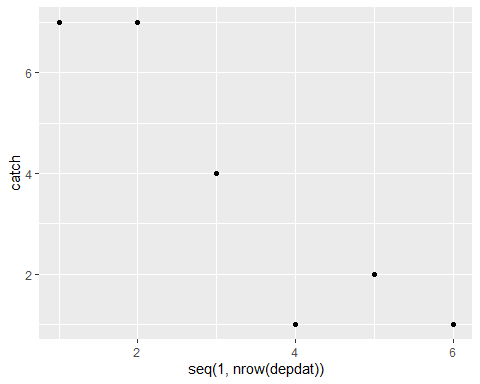
## Waiting for profiling to be done...

## estimates 2.5% 97.5%  
## a 1.494194e+01 6.636636e+00 3.983782e+01  
## b 3.474890e-05 1.722387e-05 7.397983e-05

#Visualize the model fit  
#Range of spawning stock  
x <- seq(from=min(ChinookSR$spawners),   
 to = max(ChinookSR$spawners),   
 by= 10000)  
  
#Predict recruitment from model fit above  
pR<- rckr(x, a=coef(srR))  
#combine in a data frame  
CombSR <- data.frame(x = x, pR = pR)  
  
#Plot predictions with raw data  
ggplot() +  
 geom\_point(aes(x = CombSR$x, y = CombSR$pR),   
 color = "orange", size = 3) +  
 geom\_point(aes(x = ChinookSR$spawners,   
 y = ChinookSR$recruits)) +  
 xlab("Stock size") +   
 ylab("Predicted recruitment")



#Population Estimates from Depletion Data#######  
  
#Leslie Method  
#C\_i/f\_i = qN0 - q(K\_i-1)  
  
#C\_i=catch for sample i  
#f\_i=fishing effort for sample i  
#q=catchability coefficient  
#N0=initial abundance  
#k\_i-1=cumulative catch prior to sample i  
  
#Essentially a linear regression problem  
  
#Build a data frame with capture data  
depdat <- data.frame(catch = c(7,7,4,1,2,1),  
 effort = c(10,10,10,10,6,10)) %>%  
 mutate(cpe = catch/effort, K = pcumsum(catch))  
  
#Plot catch data  
ggplot(depdat, aes(x = seq(1,nrow(depdat)), y = catch)) +  
 geom\_point()



lm2 <- lm(cpe ~ K, data=depdat)  
#extract coefficients  
(cf1 <- coef(lm2))

## (Intercept) K   
## 0.78643157 -0.03019312

#Calculate N0  
#C\_i/f\_i = qN0 - qK\_i-1  
  
#recall intercept = qN0  
#q is also the slope  
#N0 = qNO/-q or intercept/-slope  
(q\_hat <- -cf1[["K"]])

## [1] 0.03019312

(N0\_hat <- cf1[["(Intercept)"]] / q\_hat)

## [1] 26.04672

#k-pass removal estimates require equal catchability and equal effort:  
#removal(catch, method)  
#Carle Strub (default) weighted k-pass estimator  
#Burnham is a likelihood based estimator used in Microfish software (Van Deventer 1989)  
catch <- c(187, 77, 35, 5)  
pr1 <- removal(catch, method = "CarleStrub")  
  
#Extract estimates with 95% confidence intervals  
cbind(summary(pr1), confint(pr1))

## Estimate Std. Error 95% LCI 95% UCI  
## No 310.0000000 3.1830289 303.7613780 316.2386220  
## p 0.6204082 0.0272384 0.5670219 0.6737944

#The previous example can be applied to a single site.  
#Multiple sites can be estimated by using the streamlined code below  
  
#Data from three sites  
catch2 <- data.frame(sta = c("SC10","SC11","SC12"),  
 p1 = c(19,75,20), #Pass 1  
 p2 = c(14,19,11), #Pass 2  
 p3 = c(9,5,3) ) #Pass 3  
catch2

## sta p1 p2 p3  
## 1 SC10 19 14 9  
## 2 SC11 75 19 5  
## 3 SC12 20 11 3

#Use the apply function to generate population estimate at all sites  
#apply(array or matrix, Margin, function,   
# and just.est)  
#MARGIN = 1 indicates function is applied over rows  
#MARGIN = 1 indicates function is applied over columns  
#The data frame has one site for each row  
res <- apply(catch2[,-1],   
 MARGIN = 1,   
 FUN = removal,   
 just.est=TRUE)  
  
(res <- data.frame(sta=catch2$sta, t(res) ) )

## sta No No.se No.LCI No.UCI p p.se p.LCI  
## 1 SC10 55 11.727984 32.01357 77.98643 0.3716814 0.12613965 0.1244522  
## 2 SC11 100 1.470384 97.11810 102.88190 0.7557252 0.04549002 0.6665664  
## 3 SC12 36 2.507203 31.08597 40.91403 0.5964912 0.10295277 0.3947075  
## p.UCI  
## 1 0.6189106  
## 2 0.8448840  
## 3 0.7982749

#Exercise 4